

# Study on Fuzzy CMAC and Its Equivalence to Neural Fuzzy Networks

Shun-Feng Su and Shu-An He

## Abstract

**The Cerebellar Model Arithmetic Controller (CMAC) is an intelligent controller like neural networks. Different from neural networks, CMAC can be regarded as one kind of “table-look-up” learning. Research shows that by including the fuzzy concept into the cell structure of CMAC, the accuracy can be significantly improved. Such an approach is called Fuzzy CMAC (FCMAC). In this study, it will be shown that FCMAC is very similar to the Neural-Fuzzy Networks (NFN) under certain conditions. In fact, if the locations of fuzzy rules in NFN are arranged to be the same as the locations of the cells in FCMAC, then we can say that NFN and FCMAC are equivalent provided that the simplified TSK fuzzy model is considered in NFN. This paper is to report our study about the learning performance comparison between FCMAC and NFN. It can be found that because FCMAC has more than one layer to improve the association while retrieving data, the generalization capability is better than that of NFN. From simulation, it is evident that the FCMAC model has faster error convergent speed and better noise tolerance.**

**Keywords:** *Fuzzy CMAC, Neural Fuzzy Networks, Equivalence.*

## 1. Introduction

The Cerebellar Model Arithmetic Controller (CMAC) is a neurological model often used as an adaptive system for human-like nonlinear controller design. The model of CMAC was first proposed in [1], [2] and employed in various applications [3]-[11]. The learning mechanism of CMAC is to imitate the cerebellum of human being. Actually, the CMAC is also referred to as an associative neural network in which a small subset addressed by the input space determines output instantaneously, and this operation has been regarded as a special way of table-look-up leaning mechanism. Several advantages regarding CMAC including local generalization and

rapid learning convergence have been demonstrated in the literature [12], [13]. The basic operational concept of conventional CMAC can be found in [14], [15]. Early studies on CMAC are focused on research issues such as improvement of the CMAC topology structure [16]-[18], selection of learning parameters [19], convergence property of CMAC [12], [13] and modification of learning algorithms [14].

Another group of study on CMAC is to improve the accuracy capability by including the fuzzy concept into the cell structure of CMAC. This CMAC model is called Fuzzy-CMAC (FCMAC) [15], [20]-[22]. As expected, the fuzzy kind of interpretation capability indeed can increase the accuracy of the representation of the stored knowledge in FCMAC. In this study, it will be shown that FCMAC is very similar to the Neural-Fuzzy Networks (NFN) [23]-[31] under certain conditions. NFN is basically a fuzzy model in which the learning mechanism used in neural networks is employed. NFN has been shown to have nice modeling capability in various applications [23], [24], [26]-[29]. In fact, due to the fulfillment of the minimum disturbance principle [26], [30] in fuzzy systems, neural fuzzy systems are always claimed to have better modeling performance than neural networks do. In fact, if a FCMAC model has only one layer for each dimension, then it will be exactly the same as NFN. Then, an interesting question arises: is there any advantage for using FCMAC while the layer number is not 1? If the answer is positive, then another question may be asked: how many layers should FCMAC have? Such questions are investigated in our study and the results of simulations are reported in this paper.

In Section 2, the concept of conventional CMAC and of FCMAC is given. In Section 3, we briefly introduce the structure of NFN. Besides, the similarity between FCMAC and NFN is pointed out and discussed in the section. Experiments are conducted to demonstrate the learning performances of FCMAC and of NFN. Three aspects are investigated in our study. The first aspect is to discuss the learning properties of FCMAC and of NFN. The second aspect is to check their generalization capability. Finally, we also study the issue of the number of blocks or of layers in FCMAC. The results of the first two aspects are reported in Section 4. In Section 5, the performance analysis about the issue of the number of blocks or of layers in FCMAC is shown. Finally, conclusions are given in Section 6.

---

Corresponding Author: Shun-Feng Su. is with the Department of Electrical Engineering, National Taipei University of Technology, Taiwan (also with National Taiwan University of Science and Technology).  
E-mail: su@orion.ee.ntust.edu.tw

### 2. Cerebellar Model Arithmetic Controller

CMAC is a neurological model that is often used as an adaptive system for human-like nonlinear controller design. CMAC can learn functions for many degrees of freedom and products outputs by referring to a table rather than by solving mathematical equations. In this section, we shall introduce the concept of conventional CMAC and the concept of FCMAC, which include the fuzzy concept into CMAC to have better accuracy.

CMAC can be regarded as a “table-look-up” scheme. The basic structure of CMAC is shown in Figure 1. In general, two phases of operations are performed in a CMAC algorithm, the output producing phase and the learning phase. The output phase contains three parts. First, the input data are mapped into the learning space encoded by  $\mathbf{s}_k = (s_1, s_2, \dots, s_{Nd})$  where  $N_d$  is the dimension number. Secondly, the mapped states of every dimension,  $s_1, s_2, \dots, s_{Nd}$  (another name is location in the literature) in the learning space is transformed into the association cell (or called hypercube) indices,  $b_1(\mathbf{s}_k), b_2(\mathbf{s}_k), \dots, b_{n_c}(\mathbf{s}_k)$ . As shown in Figure 1, the learning space  $\mathbf{s}_k$  uses indices  $b_1(\mathbf{s}_k), b_2(\mathbf{s}_k), \dots, b_{n_c}(\mathbf{s}_k)$  to address the stored weight  $w_1, w_2, \dots, w_{n_c}$  in cells.

Finally, the output is computed as the product of the memory index vector  $\Gamma$  and the weight vector  $\mathbf{w}$ , where  $\Gamma = [b_1(\mathbf{s}_k) \ b_2(\mathbf{s}_k) \ \dots \ b_{n_c}(\mathbf{s}_k)]^T$  and  $\mathbf{w} = [w_1 \ w_2 \ \dots \ w_{n_c}]^T$ . Note that different states in the learning space should be mapped into at least one different cell index, but the association cell indices mapped by similar states may share some common cells for adjacent states. Thus, CMAC possesses the generalization ability. The learning phase is to update the addressed weights in memory cells according to the error between the desired output and the actual output.

It is easy to find that the accuracy of convention CMAC is not good enough if the number of cells is not large enough. However, if more cells are used, the training patterns must also be sufficient to cover all cells; otherwise, those unlearned cells will seriously affect the modeling accuracy. It is nature to consider the incorporation of the fuzzy concept to improve the accuracy of CMAC. Various fuzzy CMAC approaches of including fuzzy set theory into CMAC have been proposed in the literature [32], [33]. The effects of FCMAC indeed are significant as reported in the literature.

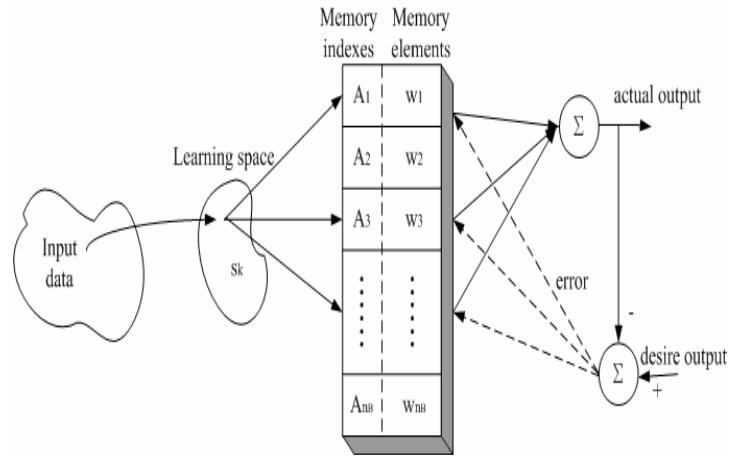


Figure 1 Basic operational concept of CMAC.

Fuzzy set theory was introduced by Zadeh in 1965 [34] and has been successfully employed in a variety of fields in recent years. It is initially proposed as a tool to model inaccurate information inherent in human reasoning. In FCMAC, the concept of fuzzy is incorporated into CMAC models so as to provide more accurate resolution. The mapping process of FCMAC is almost the same as that of CMAC except that some fuzzy mechanisms are incorporated into the mapping process. We shall introduce them in the following:

*Step 1.* (Input data  $\rightarrow$  Learning space): The mapping process of this step is like that of convention CMAC. But in FCMAC, the fuzzy set is incorporated into this step. The fuzzy membership function is included into the blocks and it is to say blocks are fuzzified [33]. Figure 2 is an example. This kind of approach is used in this paper. Due to the inclusion of membership functions into blocks,  $b_j$  is not longer either 1 or 0 but a value between 0 and 1.

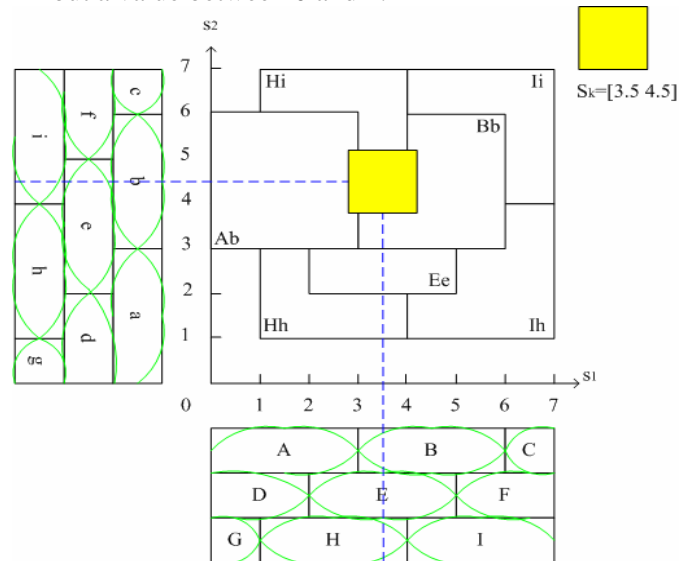


Figure 2 A 2-D FCMAC structure.

*Step 2.* (Learning space  $\rightarrow$  Memory index): At step 1, FCMAC uses the fuzzification method of the fuzzy system as its input addressing scheme. The entry of the memory index vector  $\Gamma$  is obtained by combining (producing together) those membership degrees of the associated blocks.

*Step 3.* (Memory index  $\rightarrow$  Output): This step is all the same as that of convention CMAC. The output is described as

$$y = \mathbf{w}^T \Gamma = \sum_{j=1}^{n_H} w_j(i) \times b_j(s_k) \quad (1)$$

*Step 4.* (Output  $\rightarrow$  Weight update): This step is similar to convention CMAC, but some differences exist. The weight updating algorithm is described as

$$w_j(i+1) = w_j(i) + \frac{\alpha}{n_L} b_j(s_k) \left[ y_d - \sum_{j=1}^{n_H} w_j(i) \times b_j(s_k) \right] \quad (2)$$

In this kind of FCMAC, fuzzy sets are employed to fuzzify the index of the addressed cells. There are also other types of CMAC or fuzzified CMAC methods discussed in the literatures. For example, in [35], the authors proposed a locally weighted regression technique (LWR\_CMAC) to organize the collected weights and make the output of the learning module differentiable so that the error can be back-propagated to determine the responsibility of each input component. In [36], the authors used an optimal weight smoothing method to cancel a disturbance input in the control loop. There is another approach, Hierarchical CMAC mentioned in [38]. The Hierarchical CMAC consists of a self-organizing input space module and a hierarchical network to facilitate the fuzzification process. Besides a recurrent CMAC is proposed in [39], [40].

Traditionally, triangular membership functions are used in FCMAC. A tip about how to efficiently employ triangular membership functions is also presented in [15]. In this study, we intend to investigate the comparison between FCMAC and NFN. In order to have the same learning effects, the *backpropagation* (BP) learning algorithm, which is often used for NFN, will also be used. It is noted that if triangular membership functions are used, it is difficult to apply the BP learning algorithm because the algorithm needs to take derivative of any functions used and triangular functions have points that are not differentiable. Thus, in this study, Gaussian functions are used as fuzzy membership functions both in FCMAC and in NFN. In fact, such a CMAC model is often called Gaussian-CMAC (GCMAC) [37]. In this paper, we still refer it as FCMAC.

### 3. NFN and Its Similarity to FCMAC

In our study, we found that FCMAC and NFN have a similar structure if certain conditions are satisfied. In this section, we shall introduce the architecture and the learning procedure of NFN [23]-[30]. Figure 3 shows a general architecture for NFN [26], [27], [41], [42]. It is a network with four layers of neurons. They are the input layer, the linguistic term layer, the rule layer and the output layer. In our implementation, the numbers of neurons in hidden layers are  $n_B$  and  $n_H$ . The vector  $s_k$  is mapped into the linguistic term layer and yields the output as  $b_u(s_k) = e^{-(s_k - m_u)^2 / \sigma_u^2}$ ,  $u = 1, 2, \dots, n_B$ . Then, the outputs of the linguistic term layer are sent to the rule layer and its output is computed as  $w_i \cdot b_i$ , where  $w_i$  is the weight of neuron  $i$  in the rule layer and

$b_i(s_k) = b_u(s_k) \times b_v(s_k) = e^{-(s_k - m_u)^2 / \sigma_u^2} \times e^{-(s_k - m_v)^2 / \sigma_v^2}$ ,  $i = 1, 2, \dots, n_H$ . Here, the fuzzy rule structure used is the simplified TSK fuzzy model [43], in which the consequence parts of the fuzzy rules are crisp values (i.e.,  $w_i$ ). Let the weight vector be  $\mathbf{w} = [w_1 \ w_2 \ \dots \ w_{n_H}]^T$  and the relative degree vector  $\Gamma$  be  $\Gamma = [b_1(s_k) \ b_2(s_k) \ \dots \ b_{n_H}(s_k)]^T$ . Then, the overall output is [41], [42]:

$$y = \mathbf{w}^T \Gamma = \sum_{j=1}^{n_H} w_j(i) \times b_j(s_k). \quad (3)$$

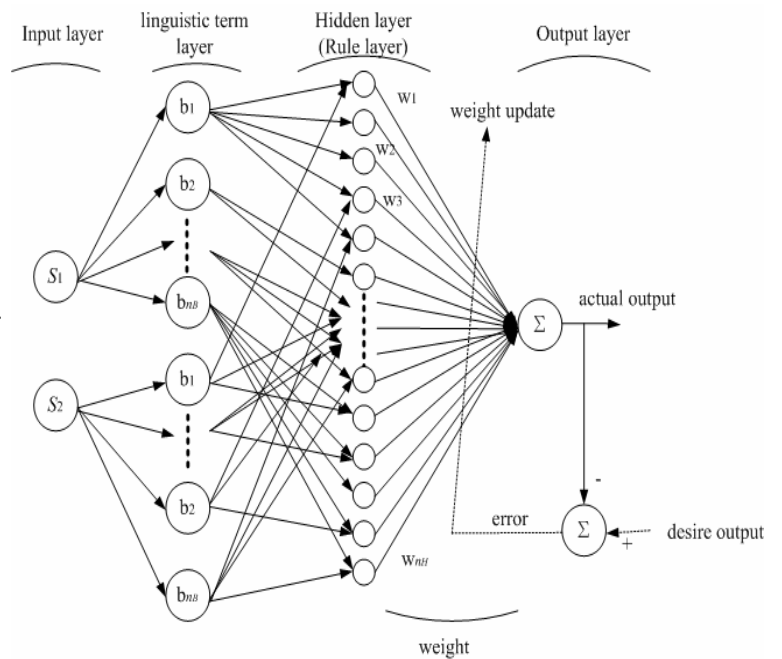


Figure 3 A general Neural Fuzzy model.

It can be found that Equations (1) and (3) are the same. Now, if the same learning algorithm (BP) is employed, the resultant systems will be the same after training. In other words, if the locations of fuzzy rules in NFN are

arranged to be the same as the locations of the cells in FCMAC, then we can say that NFN and FCMAC are equivalent provided that the simplified TSK fuzzy model is used in NFN. Of course, usually the locations of fuzzy rules in NFN are simply spread out for each variable independently. On the other hand, FCMAC has the "layer" structure in each dimension. Therefore, if a FCMAC model has only one layer for each dimension, then it will be exactly the same as NFN. Then, an interesting question arises: is there any advantage for using FCMAC while the layer number is not 1? If the answer is positive, then another question may be asked: how many layers should FCMAC have? Such questions are investigated in our study and the results of simulations are reported in the following sections.

In the literature, there are two different ways of tuning parameters in the consequences of a TSK fuzzy model. They are the recursive least square method [23], [31] and the traditional BP learning algorithm [24]-[26]. From our early study [26], the performances of those two approaches are not significantly different. In fact, due to the use of the simplified TSK fuzzy structure, there is only one unknown parameter for one rule in the recursive least square method. Thus, while the correlation between rules is neglected, it becomes a simple update algorithm and is similar to the perceptron learning [44], in which the error is directly used as the error signal in the learning process. This perceptron learning algorithm is simple and intuitive [42]. In this study, in addition to the BP learning algorithm, we also consider this learning algorithm in our experiments for comparison.

#### 4. Comparison for FCMAC and NFN

In this section, we shall report our analysis on the comparison between FCMAC and NFN. Experiments are conducted to illustrate the comparison. In the study, two functions are used as test instances; they are  $\sin(\pi x) \cdot \cos(\pi y)$ ,  $x, y \in [-1, 1]$  (marked as example *i*) and  $(x^2 - y^2) \sin(5y)$ ,  $x, y \in [-1, 1]$  (marked as example *ii*). In this implementation, 1600 (40×40) training data patterns are obtained with equally sampling on the  $x$  and on the  $y$  axes, respectively. It is known that the performance of a learning scheme can be evaluated in two folds. They are the training performance and the testing performance. We shall consider them separately as follows.

For the training performance, the convergent behavior is concerned. Usually, the convergent behavior is illustrated by learning histories, which will show training errors and the numbers of training epochs. In our comparison, two cases are considered. The first one is to

have the same or approximately the same number of cells or rules and the other one is to have the same resolution in memories (i.e., blocks or linguistic terms). Also, two situations are considered; sparse or condensed blocks situations. Additionally, as mentioned earlier, there are two learning approaches, *perceptron* and *backpropagation* (BP), used in NFN. Both learning algorithms are employed in our study.

First, we consider the sparse cases. In this situation, we choose a 2-D FCMAC model with two layers in each dimension, 5 blocks for each layer, and 50 cells totally. To satisfy the condition of approximately the same number of cells, NFN is selected to have 7 membership functions in each dimension and then there are 49 rules. To have the same resolutions in memories, a NFN with 10 membership functions in each dimension and there are 100 rules. The learning histories of various models for those two examples are obtained and shown in Figures 4 and 5, respectively. The mean square errors (MSEs) of those training processes are all tabulated in Table 1. In example *i*, it can be found that the MSEs of FCMAC and of NFN with the perceptron learning can converge to a similar error level in the same or approximately the same number of cells or rules. But, the convergent speed of FCMAC is faster than that of NFN. But, in example *ii*, the error of FCMAC is bigger than that of NFN and the convergent speed of FCMAC is not faster than that of NFN. However, under the same resolution, NFN has a less convergent error than FCMAC does, but it needs more rules. Another interesting phenomenon is that NFN with the BP learning needs more time to converge than NFN with the perceptron learning does.

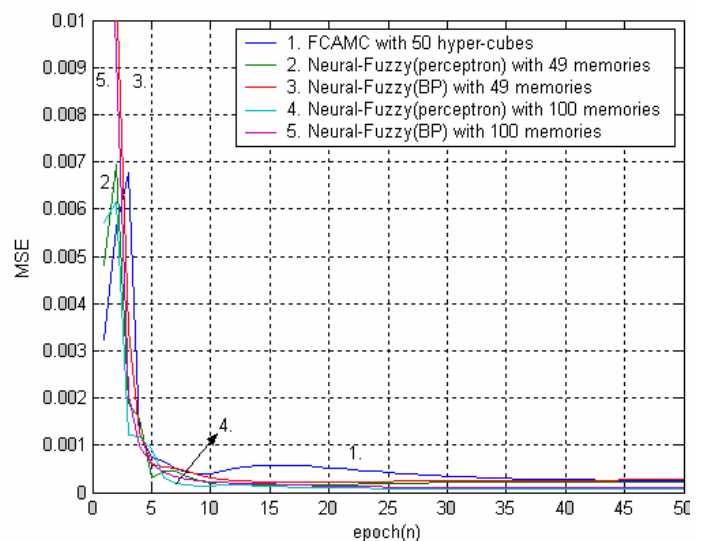


Figure 4 The learning MSE of these simulations for example *i* in sparse cases.

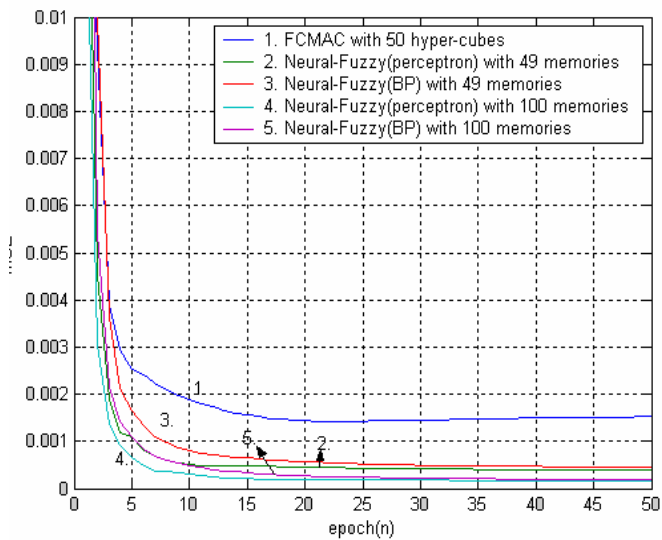


Figure 5 The learning MSE of these simulations for example *ii* in sparse cases.

Table 1 The learning MSE of these simulations for sparse cases.

		epoch						
		$n_H$	1 <sup>st</sup>	10 <sup>th</sup>	20 <sup>th</sup>	30 <sup>th</sup>	40 <sup>th</sup>	50 <sup>th</sup>
<i>Example i.</i> $\sin(\pi x) \cdot \cos(\pi y)$ , $x, y \in [-1, 1]$								
FCMAC		50	3.2	0.4	0.5	0.3	0.2	0.2
NFN (per)		49	4.8	0.2x	0.2	0.2	0.2	0.2
		100	5.7	0.1	0.09	0.07	0.07	0.07
NFN (BP)		49	14	0.3	0.2	0.2	0.2	0.2
		100	17	0.4	0.1	0.1	0.09	0.08
<i>Example ii.</i> $(x^2 - y^2) \sin(5y)$ , $x, y \in [-1, 1]$								
FCMAC		50	26	1.8	1.4	1.4	1.4	1.4
NFN (per)		49	18	0.5	0.45	0.41	0.4	0.4
		100	13	0.29	0.18	0.17	0.16	0.15
NFN (BP)		49	32	0.8	0.5	0.49	0.46	0.45
		100	26	0.48	0.27	0.22	0.19	0.18

Next, the condensed cases are considered. Consider a 2-D FCMAC model which has two layers in each dimension and with 16 blocks for each layer, and then there are 512 cells totally. For NFN, to satisfy approximately the same number of cells, the 2-D NFN used has 23 membership functions in each dimension and there are 529 rules. To have the same resolution, NFN has 32 membership functions in each dimension and there are 1024 rules. The results are shown in Figures 6 and 7 and the MSEs are listed in Table 2.

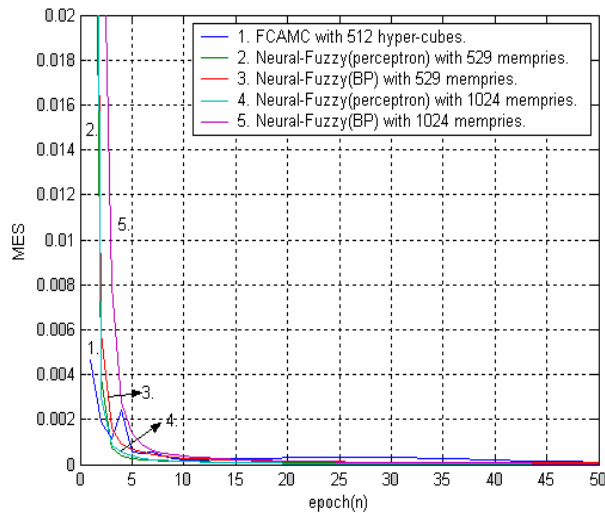


Figure 6 The learning MSE of these simulations for example *i* in condensed cases.

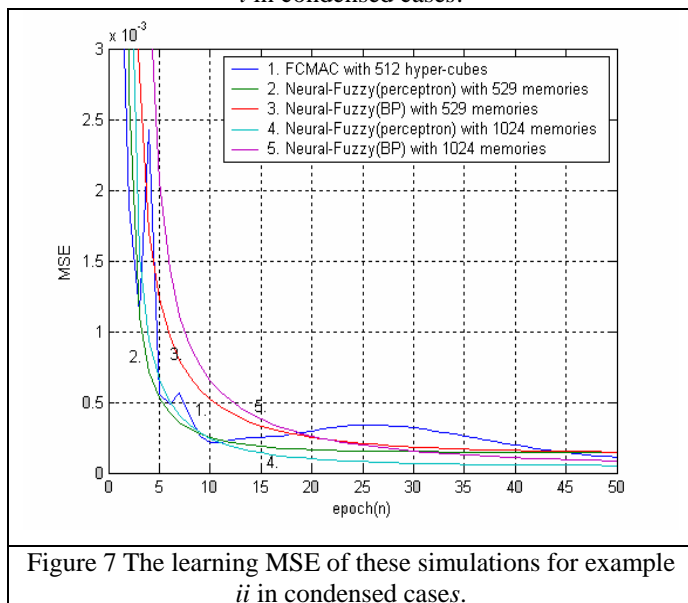


Figure 7 The learning MSE of these simulations for example *ii* in condensed cases.

From those results, it can be found that when the structure of the fuzzy sets used (or say the blocks size) is condensed, the obtained results are more accurate than that of the sparse cases no matter which modeling approach is used. From those experiments, it is evident that FCMAC has the property of faster convergence and NFN has the property of more accurate learning.

Now, the testing performance is considered or the generalization capability is concerned. It is to check whether the learned system can interpret unlearned patterns well. In addition to 1600 training data patterns, a new set of data must be used for testing. In this study, another 10000 data are arbitrarily generated within the considered domain, [-1 1]. In our study, two conditions are considered. The first one is to consider noise-free data and the other is to consider data with noise of which the mean and the standard deviation are 0 and 0.1,

respectively. In each simulation both sparse and condensed blocks are employed to check their effects. The simulation results are shown in Tables 3 and 4.

**Table 2** The learning MSE of these simulations for condensed cases.

		epoch					
	$n_H$	1 <sup>st</sup>	10 <sup>th</sup>	20 <sup>th</sup>	30 <sup>th</sup>	40 <sup>th</sup>	50 <sup>th</sup>
<i>Example i.</i> $\sin(\pi x) \cdot \cos(\pi y)$ , $x, y \in [-1, 1]$							
FCMAC	512	3.2	0.4	0.5	0.3	0.2	0.2
NFN (per)	529	4.6	0.2	0.2	0.2	0.1	0.1
	1024	47	0.1	0.09	0.08	0.08	0.08
NFN (BP)	529	90	0.1	0.05	0.03	0.03	0.03
	1024	92	0.3	0.1	0.1	0.1	0.08
<i>Example ii.</i> $(x^2 - y^2) \sin(5y)$ , $x, y \in [-1, 1]$							
FCMAC	512	4.6	0.2	0.2	0.2	0.1	0.1
NFN (per)	529	47	0.1	0.09	0.08	0.08	0.08
	1024	90	0.1	0.05	0.03	0.03	0.03
NFN (BP)	529	92	0.3	0.1	0.1	0.1	0.08
	1024	140	0.3	0.1	0.09	0.06	0.04

Table 3 The testing MSE for noise-free data.

		FCMAC		NFN			
	$n_H$	50	512	49	100	529	1024
<i>Example i.</i> $\sin(\pi x) \cdot \cos(\pi y)$ , $x, y \in [-1, 1]$							
MSE	0.6	0.6	perceptron				
			0.6	0.06	0.6	0.07	
			backpropagation				
			0.3	0.06	0.6	0.07	
<i>Example ii.</i> $(x^2 - y^2) \sin(5y)$ , $x, y \in [-1, 1]$							
MSE	0.45	0.15	perceptron				
			0.5	0.12	0.12	0.11	
			backpropagation				
			0.38	0.12	0.11	0.11	

From the results, it is obvious that the MSEs for noise-free data are less than that for noisy data. In both the sparse and condensed blocks situations, it can be found that the MSE of FCMAC is equal or less than that of NFN with the perceptron learning in the same or approximately the same number of cells or rules

situation. However, in the same resolution of memories, the MSE of FCMAC is bigger than that of NFN with the perceptron approach. Additionally, if NFN is trained by the BP learning, the MSE of NFN will be less than that of FCMAC. Thus, we have reached a conclusion that FCMAC has better noise tolerance than NFN with the perceptron learning does. It is because FCMAC possesses the association characteristics. But, when the BP learning is employed for NFN, due to the optimization process of BP, the noise tolerance can become the best of all three approaches.

**Table 4** The testing MSE for noisy data.

		FCMAC		NFN			
	$n_H$	50	512	49	100	529	1024
<i>Example i.</i> $\sin(\pi x) \cdot \cos(\pi y)$ , $x, y \in [-1, 1]$							
MSE	0.7	0.7	perceptron				
			0.8	0.3	0.87	1.5	
			backpropagation				
			0.5	0.2	0.7	1.2	
<i>Example ii.</i> $(x^2 - y^2) \sin(5y)$ , $x, y \in [-1, 1]$							
MSE	0.5	0.82	perceptron				
			0.53	0.38	0.95	0.15	
			backpropagation				
			0.46	0.32	0.83	0.12	

### 5. Numbers of Block and of Layer in FCMAC

From the above simulation, it is evident that FCMAC can have better generalization capability. Then, another question arises: whether such advantages can be further improved if more layers are used? In this section, given a fixed number of cells, how many blocks and layers are required so that the learning performance is the best? It is easy to see that for a fixed number of cells, if there are more layers in each dimension of the 2-D CMAC model, then fewer blocks will be divided for each layer and vice versa. In this simulation, 144 hyper-cubes are considered and then there are five possible block-layer combinations: 1) 1 layer and 12 blocks for each layer. 2) 4 layer and 6 blocks for each layer. 3) 9 layer and 4 blocks for each layer. 4) 16 layer and 3 blocks for each layer. 5) 36 layers and 2 blocks for each layer.

The same learning constant is used for the first three conditions. But experiments with conditions 4 and 5 need too many epochs to converge. Thus, a slight bigger value is used for the learning constant in conditions 4

and 5, and it does not have much influence on the experiment results. After 100 training epochs, the learned systems are shown in Figures 8 and 9 for example *i* and example *ii*, respectively, and the MSEs of these conditions are listed in Table 5. In above simulations, it is found that the error convergence level of condition 5 is the worst and that of condition 4 is the next worst. In other words, when the number of layers increases, even though the association properties can have more complicated forms and supposedly, the generalization capability can be further improved. Nevertheless, when the layer number increases, the learning task becomes complicated too. In other words, in each learning iteration, more cells are involved for a training pattern and thus more cells are needed to be modified for a pattern. It is easy to see that it violates the minimum distribution principle [26], [30]. As a consequence, the learning becomes inefficient when the number of layers increases and then the learning performance is not good enough as expected. It is observed that the error convergence levels for conditions 1 and 2 are approximately the same. It means the learning complexity is still tolerable for condition 2. Also it should be noted that during the 10<sup>th</sup> to the 20<sup>th</sup> epochs, the error convergence speed of condition 2 is faster than that of condition 1, and then they have approximately the same convergent error, after the 60<sup>th</sup> epoch.

Next, the testing performances for these five conditions are considered. The same 10000 data are used as testing data and 100 learning epochs are conducted to ensure the convergence of the learning. Again, two sets of testing patterns with noise-free and with noise of which the mean and the standard deviation are 0 and 0.1 are considered. The MSEs of those simulations are tabulated in Tables 6 and 7. From Tables 6 and 7, the testing errors of condition 1 and of condition 2 are the least, and condition 5 is still the worst of these five conditions. It is noted that in Table 6, the case with noise-free training data, the MSE of condition 1 is less than that condition 2. But from Table 7, the case with noisy data, the MSE of condition 2 is less than that of condition 1. Thus, we can say the noise tolerance of condition 2 is better than condition 1. This coincides with the conclusion obtained in the previous section.

### 6. Conclusions

This study discusses the learning performances of FCMAC and of NFN. Intuitively, FCMAC is like NFN but with more than one layers. Since FCMAC has more than one layer to improve the association while retrieving data, the generalization capability is better than that of NFN. Thus, it is anticipated that FCMAC

can have good learning speed and nice noise tolerance. Through these simulations conducted in our study, FCMAC indeed have those properties as we expected. As for the accuracy, the performance of FCMAC is equal to or worst than that of NFN. The reason is that when more layers are used, the learning task become complicated. Also due to more cells being involved in the learning, the minimum distribution principle states that such a learning process will be less efficient.

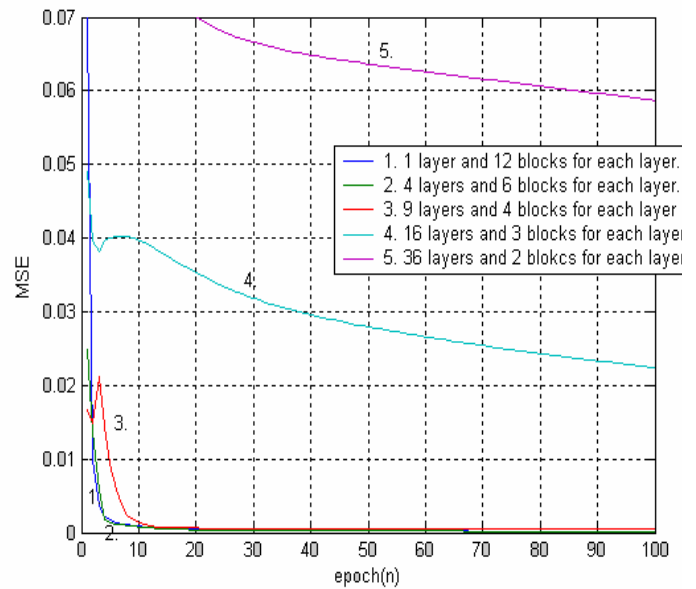


Figure 8 The learning MSE of these FCMAC models with different structures for example *i*.

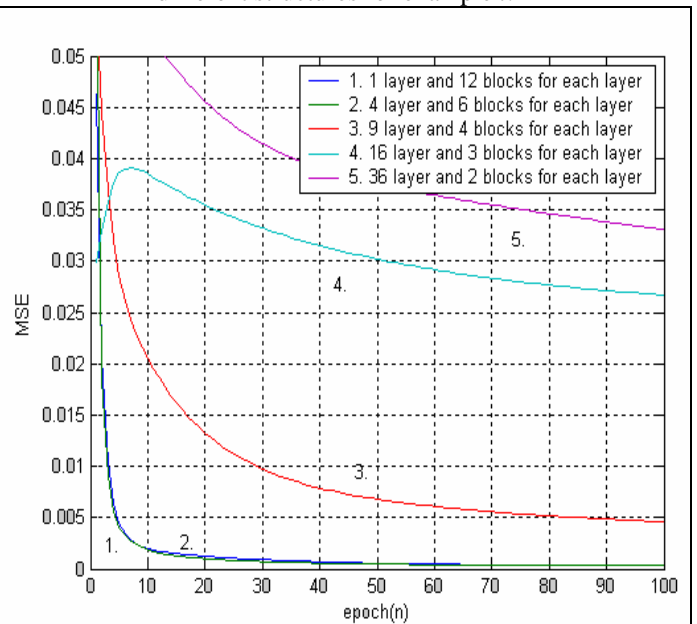


Figure 9 The learning MSE of these FCMAC models with different structures for example *ii*.

Table 5 The learning MSE at certain epochs in these FCMAC

models with different structures.

	L #	B #	epoch						
			1 <sup>st</sup>	10 <sup>th</sup>	20 <sup>th</sup>	40 <sup>th</sup>	60 <sup>th</sup>	80 <sup>th</sup>	100 <sup>th</sup>
Example i. $\sin(\pi x) \cdot \cos(\pi y)$ , $x, y \in [-1, 1]$									
1	1	12	70	0.8	0.47	0.28	0.21	0.18	0.15
2	4	6	25	0.9	0.4	0.25	0.21	0.17	0.14
3	9	4	20	3	1.73	1.5	0.5	0.48	0.48
4	16	3	48	39	35	29	26	24	22
5	36	2	99	79	70	64	62	60	58
Example ii. $(x^2 - y^2) \sin(5y)$ , $x, y \in [-1, 1]$									
1	1	12	58	1.9	1.2	0.57	0.4	0.3	0.3
2	4	6	55	1.9	0.9	0.7	0.5	0.3	0.3
3	9	4	60	23	14	9	7	5	4.9
4	16	3	26	38	35	31	29	27	26
5	36	2	34	51	46	39	36	31	33

**Table 6** The testing MSE for these five conditions with noise-free data.

		error $\times 10^{-3}$				
		1	2	3	4	5
Layer		1	4	9	16	36
Blocks		12	6	4	3	2
MSE	Example i. $\sin(\pi x) \cdot \cos(\pi y)$ , $x, y \in [-1, 1]$	0.09	0.12	0.769	32	290
	Example ii. $(x^2 - y^2) \sin(5y)$ , $x, y \in [-1, 1]$	0.177	0.234	3.7	36.1	72

**Table 7** The testing MSE for these five conditions with noisy data.

		error $\times 10^{-3}$				
		1	2	3	4	5
Layer		1	4	9	16	36
Blocks		12	6	4	3	2
MSE	Example i. $\sin(\pi x) \cdot \cos(\pi y)$ , $x, y \in [-1, 1]$	0.3	0.22	0.8	33	291
	Example ii. $(x^2 - y^2) \sin(5y)$ , $x, y \in [-1, 1]$	0.378	0.324	3.8	37	73

**References**

[1] J. S. Albus, "A new approach to manipulator control: the cerebellar model articulation controller (CMAC)," *ASME Journal of Dynamic Systems, Measurement, and Control*, pp. 220-227, 1975.

[2] J. S. Albus, "Data storage in the cerebellar model articulation controller (CMAC)," *ASME Journal of Dynamic Systems, Measurement, and Control*, pp. 228-233, 1975.

[3] F. H. Glanz, W. T. Miller, and L. G. Kraft, "An overview of the CMAC neural network," *Proceedings of 1991 IEEE Neural Networks for Ocean Engineering*, pp. 301-308, 1991.

[4] J. Hu, J. Pratt, and G. Pratt, "Stable adaptive control of a bipedal walking robot with CMAC neural networks," *Proceedings of 1999 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1050-1056, 1999.

[5] R. O. Shelton and J. K. Peterson, "Controlling a truck with an adaptive critic CMAC design," *Simulation*, vol. 58, no. 5, pp. 319-326, 1992.

[6] W. T. Miller, F. H. Glanz, and L. G. Kraft, "Application of a general learning algorithm to the control of robotic manipulators," *The International Journal of Robotic Research*, vol. 6, no. 2, pp. 84-98, 1987.

[7] Y. Iiguni, "Hierarchical image coding via cerebellar model arithmetic computers," *IEEE Transactions on Image Processing*, vol. 5, no.10, pp. 1393-1401, 1996.

[8] W. T. Miller and F. H. Glanz, "CMAC: an associative neural network alternative to backpropagation," *Proceedings of The IEEE*, vol. 78, no. 10, pp. 1561-1567, 1990.

[9] C.-M. Lin and Y.-F. Peng, "Adaptive CMAC-based supervisory control for uncertain nonlinear systems," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 2, pp. 1248-1260, 2004.

[10] S. Labiod, "Comments on "Integral variable structure control of nonlinear system using a CMAC neural network learning approach"," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 6, pp. 2421-2421, 2004.

[11] C.-P. Hung, "Integral variable structure control of nonlinear system using a CMAC neural network learning approach," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 1, pp. 702-709, 2004.

[12] C. S. Lin and C. T. Chiang, "Learning convergence of CMAC Technique," *IEEE Trans.*

- on Neural Networks, vol. 8, no.6, pp.1281-1292, 1997.
- [13] Y. F. Wong and A. Sideris, "Learning convergence in the cerebellar model articulation controller," *IEEE Trans. on Neural Networks*, vol. 3, no.1, pp.115-121, 1992.
- [14] S.-F. Su, T. Tao, and T.-H. Hung, "Credit assigned CMAC and its application to online learning robust controllers," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 33, no. 3, pp. 202-213, 2003.
- [15] S.-F. Su, Z.-J. Lee, and Y.-P. Wang, "Robust and Fast Learning for Fuzzy Cerebellar Model Articulation Controllers," *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, vol. 36, no. 1, pp. 203-208, 2006.
- [16] C. S. Lin and C. K. Li, "A new neural network structure composed of small CMACs," *Proceedings of IEEE Conference on Neural Systems*, pp. 1777-1783, 1996.
- [17] C. S. Lin and C. K. Li, "A low-dimensional-CMAC-based neural network," *Proceedings of IEEE Conference on Systems, Man and Cybernetics*, vol. 2, pp. 1297-1302, 1996.
- [18] S. H. Lane and J. Militzer, "A comparison of five algorithm for the training of CMAC memories for learning control systems," *International Federation of Automatic Control*, vol. 28, no. 5, pp. 1027-1035, 1992.
- [19] N. E. Cotter and T. J. Guillermin, "The CMAC and a theorem of kolmogorov," *Neural Networks*, vol. 5, pp. 221-228, 1991.
- [20] K.-S. Hwang and C.-S. Lin, "Smooth trajectory tracking of three-link robot: a self-organizing CMAC approach," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 28, no. 5, pp. 680-692, 1998.
- [21] J. Nie and D. A. Linkens, "FCMAC: a fuzzified cerebellar model articulation controller with self-organizing capacity," *Automatica*, vol. 30, no. 4, pp. 655-664, 1994.
- [22] Z. J. Geng and C. L. McCullough, "Missile control using fuzzy cerebellar model arithmetic computer neural networks," *Journal of Guidance, Control, and Dynamics*, vol. 20, no.3, pp. 557-565, 1997.
- [23] J. S. R. Jang, "Adaptive-network-based fuzzy inference systems," *IEEE Trans. on Systems, Man, Cybernetics*, vol. 23, no. 3, pp. 665-685, 1993.
- [24] S.-F. Su and S.-R. Huang, "Applications of model-free estimators to the stock market with the use of technical indicators and non-deterministic features," *Journal of the Chinese Institute of Engineers*, vol. 26, no. 1, pp. 21-36, 2003.
- [25] C. T. Lin and C. S. G. Lee, "Neural-network-based fuzzy logic control and decision systems," *IEEE Trans. on Computers*, vol. 40, no. 12, pp. 1320-1336, 1991.
- [26] S.-F. Su and K.-Y. Chen, "Conceptual discussions and benchmark comparison for neural networks and fuzzy systems," *Differential Equations and Dynamical Systems*, vol. 13, no. 1, pp. 35-61, 2005.
- [27] S.-F. Su and K.-Y. Chen, "Fuzzy hierarchical data fusion networks for terrain location identification problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 1, pp. 731-739, 2004.
- [28] C.-H. Lee and C.-C. Teng, "Identification and control of dynamic systems using recurrent fuzzy neural network," *IEEE Trans. on Fuzzy Systems*, vol. 8, no. 4, pp. 349-366, 2000.
- [29] S.-F. Su and F. P. Yang, "On the dynamical modeling with neural fuzzy networks" *IEEE Trans. on Neural Networks*, vol. 13, no. 6, pp. 1548-1553, 2002.
- [30] J. S. R. Jang, C.-T. Sun and E. Mizutani, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Prentice-Hall, 1997.
- [31] C. F. Juang and C. T. Lin, "An on-line self-constructing neural fuzzy inference network and its applications," *IEEE Trans. on Fuzzy Systems*, vol. 6, no. 1, pp.12-32, 1998.
- [32] J.-H., and J.-G. Hsieh, "Adaptive Tracking Control of a Class of Nonlinear Systems using CMAC Network," *J. Franklin Inst.*, vol. 333B, no. 6, pp. 861-878, 1996.
- [33] T. Tao, "The Study of Improved CMAC and Its Applications in Control and Image Process," Ph.D. Thesis, Tatung University. Taipei, Taiwan, 2003.
- [34] L. A. Zadeh, "Fuzzy Set," *Information and Control*, vol. 8, pp. 338-353, 1965.
- [35] J. Geng and T.N. Lee, "Freeway traffic incident detection using fuzzy CMAC neural networks," *IEEE World Congress on Computational Intelligence, IEEE International Conference*, vol. 2, pp. 1164-1169, 1998.
- [36] J. Nie, and D.A. Linkens, "FCMAC: a fuzzified cerebellar model articulation controller with self-organizing capacity", *Automatica*, vol. 30, no. 4, pp. 655-664, 1994.
- [37] K.-S. Hwang and C.-S. Lin, "Smooth trajectory tracking of three-link robot: a self-organizing CMAC approach," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol.

28, no. 5, Oct. 1998.

- [38] K. Zhang and F. Qian, "Fuzzy CMAC and its application," *Intelligent Control and Automation*, pp. 944-947, June 28-July 2, 2000.
- [39] Y.-F. Peng, and C.-M. Lin, "Intelligent Hybrid Control for Uncertain Nonlinear Systems Using a Recurrent Cerebellar Model Articulation Controller," *IEE Proc.-Control Theory Appl.*, vol. 151, no. 5, September 2004.
- [40] T.-F. Peng, C.-M. Lin, and W.-L. Chin, "Adaptive Recurrent Cerebellar Model Articulation Controller for Unknown Dynamic Systems with Optimal Learning-Rates," *IEEE Conference*, 2004.
- [41] H.-C. Lu and W.-J. Wang, *Design Of Cerebellar Model Articulation Controller With Fuzzy Theory And Grey Method*, Department of Electrical Engineering Tatung University, Jun. 2000.
- [42] Intelligent Automation Inc., "Fuzzy CMAC neural networks and application to flight vehicle control," *Phase I Technical Report to Ballistic Missile Defense Organization*, Sept. 1994.
- [43] T. Takagi and M. Sugeno. "Fuzzy identification of system and its applications to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. 15, no. 1, pp. 116-132, Jan. 1985.
- [44] C. T. Lin and C. S. G. Lee, *Neural Fuzzy systems : A Neural Fuzzy Synergism to Intelligent*, Prentice Hall, 1996.



**Shu-An He** received the M.S. degrees in electrical engineering, in 2006 from National Taiwan University of Science and Technology, Taiwan, R.O.C. He currently is a PhD student in Electrical Engineering, National Taiwan University.



**Shun-Feng Su** received the B.S. degree in electrical engineering, in 1983, from National Taiwan University, Taiwan, R.O.C., and the M.S. and Ph.D. degrees in electrical engineering, in 1989 and 1991, respectively, from Purdue University, West Lafayette, IN.

Dr. Su now is a Professor of the Department of Electrical Engineering, National Taiwan University of Science and Technology, Taiwan, R.O.C. He is also a Professor with the Department of Electrical Engineering, National Taipei University of Technology. He has published more than 100 refereed journal and conference papers. Dr. Su was active in related international conferences and served as various chairs in those conferences. He also served as guest editors in a couple journals. He currently is in the Editor Boards of IEEE Transactions on Systems, Man, Cybernetics, Part B, and of International Journal of Fuzzy Systems. His current research interests include computational intelligence, machine learning, robotics, bioinformatics, intelligent transportation systems, smart home, and intelligent control.